

# Allocation and Scheduling using Linear Programming in the SPIRAL Compiler

## Optimizing Compilers : Project Proposal

<http://www.ece.cmu.edu/~tstrigko/spiral/>

Frédéric de Mesmay  
fdemesma@ece.cmu.edu

Theodoros Strigkos  
strigkos@cmu.edu

### 1. Project Description

Today's compilers are usually doing register allocation and scheduling in two different passes, with the first pass imposing unnecessary constraints to the second pass. Different studies [1] have proposed simultaneous register allocation and scheduling using integer linear programming (ILP) to get an optimal solution to those problems. However, since ILP is an NP-complete problem, compilation time increases exponentially with code size, and thus, only small code segments can be scheduled this way. Nonetheless, under some conditions [2], the integer linear programming constraints can be relaxed to linear programming (LP) whose response time is polynomial instead of exponential. Our project involves implementing an LP scheduler/allocator within the SPIRAL compiler.

SPIRAL is a Carnegie Mellon University project that generates competitive code entirely autonomously for digital signal processing algorithms and other numerical kernels. When asking for a specific transform, SPIRAL will create a platform specific search space (e.g. taking into account different algorithms, simdization or parallelization) and look for the best possible implementation. In this project, we will focus our efforts on the back-end of the SPIRAL compiler, which accepts a simple language as input (loops, additions, multiplications, no pointers) and will be configured to produce fully unrolled SSE code (intrinsics or assembly).

Preliminary work by Yevgen Voronenko on Integer Linear Programming has shown potential for significant performance gains for DFTs. Indeed, in some cases, commercial compilers will spill registers in loop kernels which significantly degrades performance, whereas optimal scheduling could avoid spilling. However, the ILP algorithm's response latency is not tolerable for large block sizes, since just tens of instructions may require hours of ILP solving.

We will attempt to reformulate ILP equations to (mixed integer real) Linear Programming equations, following the methodology introduced in [2]. We hope that the relaxing of some integer constraints will allow us to use a (faster) LP solver to compile larger block sizes in reasonable time, while maintaining the optimal scheduling offered by the ILP algorithm.

### 2. Tentative Schedule

The project implementation consists of 4 stages :

- The first stage will be to set up the infrastructure, familiarize ourselves with the existing architecture of the SPIRAL compiler, as well as with the ILP module. Furthermore, we will reproduce previous results.
- For the second stage, we will study the mathematical framework of the relevant literature and try to adapt the existing model.

- The third, and main stage of the project will be to actually implement the LP scheduler/allocator within the SPIRAL compiler. We believe that we will be able to report on the beginning of this phase for the milestone.
- The final stage will be to evaluate the LP scheduler and compare its performance and time requirements against the (optimal) ILP scheduler and the scheduler of commercial compilers.

### **3. Resources Needed**

We have confirmed access to the SPIRAL software and the necessary computer clusters to run our experiments. We also need a mixed integer Linear Programming solver, and have found that certain commercial solvers offer free student licenses, which we intend to request shortly.

### **References**

- [1] C. Chang, C. Chen, and C. King. Using integer linear programming for instruction scheduling and register allocation in multiissue processors, 1997.
- [2] C. H. Gebotys and M. I. Elmasry. A global optimization approach for architectural synthesis. In *ICCAD*, pages 258–261, 1990.